Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2019/2020 **Zdeněk Poznér**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Zdeněk Poznér**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování:

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: SMART URBIDO, s.r.o.
2. Struktura závěrečné zprávy:
a. Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta
b. Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti
c. Zvolený postup řešení zadaných úkolů
d. Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe
e. Znalosti či dovednosti scházející studentovi v průběhu odborné praxe
f. Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Seznam doporučené odborné literatury:

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Zdeňka Chmelíková, Ph.D.**

Konzultant bakalářské práce: Ing. Tomáš Krempaský

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

prof. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *11. května 2020*

.....
podpis studenta

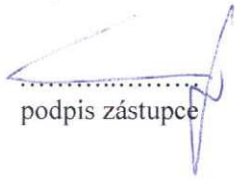
Poděkování

Rád bych poděkoval paní Ing. Zdeňce Chmelíkové, Ph.D. za odbornou pomoc a konzultaci při vytváření této bakalářské práce a také všem zaměstnancům firmy smart urbido s.r.o. za jejich trpělivost a pomoc při vývoji.

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

„Souhlasím se zveřejněním této bakalářské/diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava."

Dne: 11. května 2020



podpis zástupce

Abstrakt

Cílem této bakalářské práce bylo vytvořit multiplatformní mobilní aplikaci pro firmu smart urbido s.r.o. V této práci naleznete celý postup řešení tohoto úkolu, od samotného vybírání vhodného vývojového prostředí, kdy bylo nutné se rozhodnout, zda aplikaci vyvíjet separátně pro každou platformu či nikoli, a jazyka, přes vývoj až po samotné vydání aplikace na Google Play. V práci se také krátce seznámíte s firmou smart urbido s.r.o., s tím, co dělá a proč pro ně byla vlastní mobilní aplikace tak důležitá a také co vše mi necelý rok praxe v IT společnosti dal.

Klíčová slova

Mobilní aplikace, Angular, Android Studio, Android, iOS, Ionic, OpenStreetMap, Mapbox, Hammer.js, Chart.js, Typescript, Google Play, odborná praxe

Abstract

The purpose of this bachelor thesis was to develop a multiplatform mobile application for the smart urbido s.r.o. The purpose of this document is to summarize all the necessary things that I faced when developing an app. This includes deciding if the application will be developed on each platform separately or not, selecting the right integrated development environment as well as programming language and the publication of the app itself. It also describes the company itself, why they need a mobile solution and what did I learn there.

Key words

Mobile application, Angular, Android Studio, Android, iOS, Ionic, OpenStreetMap, Mapbox, Hammer.js, Chart.js, Typescript, Google Play, professional practice

Seznam použitých zkratek

Zkratka	Význam
CSS	Cascading Style Sheets
SCSS/SASS	Syntatically Awesome Style Sheets
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
OOP	Objektově Orientované Programování
SDK	Software Development Kit
NPM	Node Package Manager
TS	Type Script
SSHFS	Secure Shell File System
AQI	Air Quality Index
ČHMÚ	Český Hydrometeorologický Ústav
PM	Particulate Matter
AQI	Air Quality Index
IoT	Internet Of Things
iOS	iPhone Operating Systém
UI	User Interface
API	Application Programming Interface
GUI	Graphical User Interface
XML	Extensible Markup Language
FTP	File Transfer Protocolf
JSON	JavaScript Object Notation
GPS	Global Positioning System
CLI	Command Line
JS	Java Script
SSH	Secure Shell
SFTP	Secure File Transfer Protocol

Obsah

Seznam obrázků

Úvod

Tato bakalářská práce je výsledkem mého zaměstnání ve firmě urbido s.r.o., kde mi byl zadán úkol vytvořit mobilní aplikaci na míru, která bude sloužit jako reprezentace jejich IoT sekce. Praxi u firmy, místo klasické bakalářské práce, jsem si nakonec jednoduše vybral z důvodu lepší přípravy na život programátora. I když jsem na začátku vůbec nevěděl, co mě vlastně čeká, řekl bych, že má praxe byla jak pro zaměstnavatele, tak pro mě velký úspěch a obě části se naučily něco nového.

Celá práce má dohromady 7 kapitol. V první se lze dozvědět něco o zaměstnavateli, v druhé něco o problematice samotného znečištěného ovzduší. Třetí kapitola se zaměřuje na samotný vývoj a seznámení čtenáře s technologiemi, které jsem použil při tvorbě. Nejdelší čtvrtá kapitola nás pak seznamuje s procesem tvorby každé z částí aplikace, co tato část umí a k čemu jí bude následný uživatel potřebovat. Následují kapitoly o potřebném testování a ošetřování chyb a samotná část o tom, jak se aplikace vydala do širokého publika pomocí Google Play.

V neposlední řadě bych chtěl shrnout tuto bakalářskou práci jako jakýsi popis a návod, jak se vlastně taková aplikace vytváří od začátku až do konce.

1 Smart urbido s.r.o

1.1 Informace o společnosti

Urbido je společnost, která se zabývá oblastmi Smart City, eGovernmentu, IoT nebo facility managementem: provozem a údržbou, správou nemovitostí, provozu osvětlení, údržbou komunikací nebo mobiliářů. Zaměřují se se na chytré obce a inovativní společnosti.

První velký projekt této firmy byla webová aplikace určena ke správě obcí a měst. Zahrnuje 3D mapu, systém na evidenci majetku, přehled revizí a kontrol týkajících se této obce či města a nebo také hromadné SMS či maily. S tímto projektem se také přihlásili do projektu GREEN LIGHT Akcelerátor, kde získali odměnu 100 000 Kč od Moravskoslezského kraje. V neposlední řadě se také společnost začala zabývat IoT sektorem, který zahrnuje analýzu a sběr dat týkajících se kvality ovzduší České republiky.

Obrázek 1*: Logo urbido* [1]

Proč mobilní aplikace?

Tuto otázku si může klást kdekdo. Avšak v dnešním světě, kde je již většina osob připoutána ke svým mobilním zařízením, je mobilní aplikace jedna z věcí, která danou společnost může povznést do povědomí širokého okolí. Vedení společnosti si všimlo, že na trhu chybí právě aplikace, která by přehledně a přívětivě informovala uživatele o kvalitě ovzduší v jejich okolí. České aplikace, které by tuto službu prováděly sice již existují, avšak tyto aplikace jsou systémově nestabilní, jejich funkce nedostačující či uživatelsky nepřívětivé. K tomu všemu se smart urbido s.r.o. také rozhodlo vyrábět své vlastní senzory kvality ovzduší, a tak k nim doplňující se software pro mobilní zařízení byl pro úspěšný prodej vyžadován.

1.2 Zařazení do firemního prostředí

Při výběru bakalářské práce jsem si nebyl jist, kterou cestou se vydat. Na jedné straně byla možnost klasické bakalářské práce, která by mě do hloubky rozvinula v jedné specifické technologii či problematice a na druhé straně byla praxe, která by mi dala poohlédnutí po tom, jak to v reálné firmě vypadá a co se očekává od programátora na plný úvazek. Obě mají svá pro i proti, avšak nabídka firmy urbido vyvinout zcela sám mobilní řešení jejich IoT aplikace mě natolik nadchla, že jsem se rozhodl získat zkušenosti s nimi.

Při prvním setkání jsem cítil pocity určité nejistoty a nervozity plynoucí z čistého faktu, že nikdo ve firmě nemá zkušenosti s programováním profesionálních mobilních aplikací a jediné co jsem v tu chvíli věděl bylo, jak má konečné řešení přibližně vypadat. Já samotný jsem měl také jen velmi málo zkušeností s mobilními aplikacemi, avšak možnost naučit se něco, co je v dnešní době žádané mnoha firmami v okolí i ve světě nakonec překonala všechno.

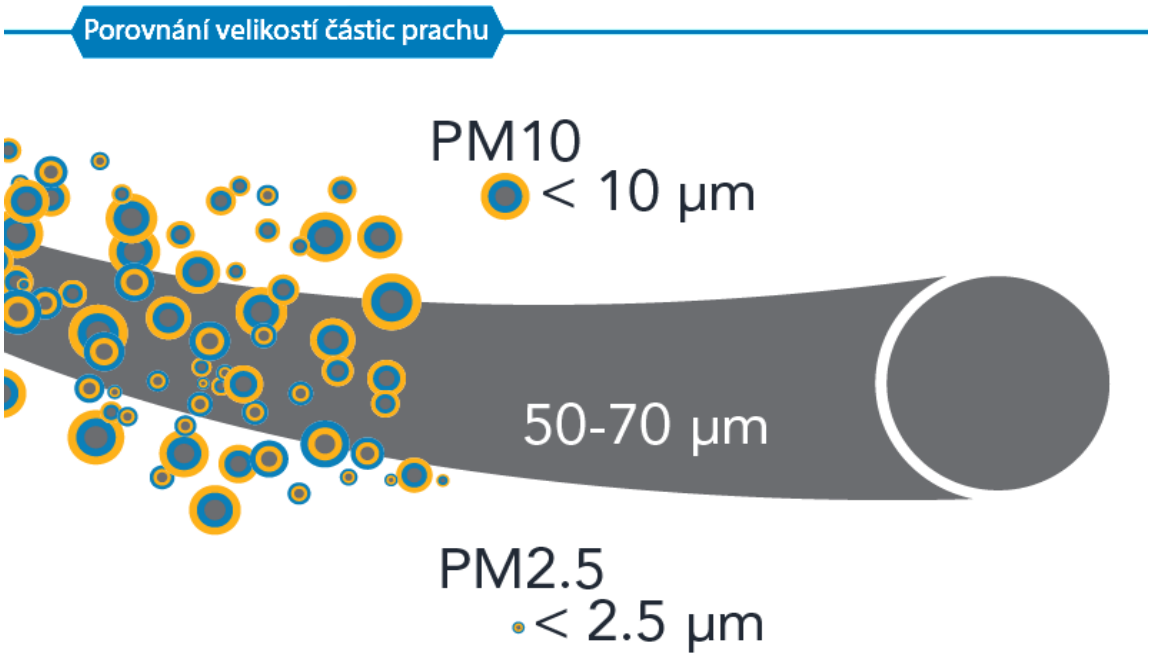
Byl jsem tedy přijat na základě referencí jednoho ze zakladatelů a přiřazen na svou pozici vývojáře mobilní aplikace. Celý startup sídlí v jedné z kanceláří Moravskoslezského inovačního centra, a tak jsem měl neustále celé vedení po ruce a všechna má řešení a návrhy jsem s nimi konzultoval přímo během implementace.

Jako výhodu vývoje o samotě bych vyzdvihnul možnou volbu technologií na základě mých nápadů, nepovinnost snažit se porozumět kódu mých potencionálních spolupracovníků a možnost naučit se porozumět vývoji aplikace od samotných začátků až po samotné vydání na trh Google Play.

2 Znečištění ovzduší

2.1 Základní informace

V dnešní době je kvalita ovzduší velice populární téma, o které se zajímá veliké spektrum lidí. Kvalita ovzduší totiž ovlivňuje život nás všech. Ať už jste prostý člověk, který nechce při venkovní procházce dýchat znečištěný vzduch či ředitel automobilky, jejíž auta musí mít právě kvůli tomuto co nejnižší emise. Jedná se tedy o částečky prachu volně poletující v ovzduší kolem nás, jež rozdělujeme do více skupin. Obecně také platí, že čím je částice menší, tím je po člověka nebezpečnější, jelikož se je schopná dostat hlouběji do lidského těla. Má aplikace má zobrazovat měření právě skupiny PM10, jež jsou částice v průměru menší jako 10 µm a PM2,5, což logicky odpovídá částicím prachu menším než 2,5 µm. Z těchto parametrů lze dále spočítat index kvality ovzduší (AQI), jež je veřejnosti bližší a má stanovené hranice pro každý stupeň nebezpečnosti situace ovzduší. Tento index bude v aplikaci zobrazován jako hlavní indikátor současné situace.



Obrázek 2*: Velikost částic prachu* [2]

2.2 Kdo jej měří

Jak už jsem napsal výše, společnost, u které má praxe probíhala, si své senzory vyrábí sama, jejich řešení je kvalitní, ale ne dostupné pro každého. Pokud vás tato problematika zajímá, můžete si najít svůj nejbližší měřící senzor na stránkách ČHMÚ, kde lze najít interaktivní mapu se všemi senzory patřícími této organizaci, jež je pravidelně aktualizována a přináší nám nová data každou celou hodinu. Tato data jsem poté společně s daty ze senzorů zaměstnavatele zpracovával v mé mobilní aplikaci, jež dostala název Urbido Air.

3 Použité technologie

3.1 Angular

Je to open source framework pro webové aplikace založený na TypeScriptu. Vyvíjený převážně Angular Teamem v Googlu. Je to poměrně nový prostředek k tvorbě frontendu ale i backendu webových aplikací. Za pouhé 3 roky jeho existence, ho převzalo mnoho světových firem jako hlavní zdroj tvorby front endu jejich webových aplikací. Z nejznámějších to je například Samsung, PayPal, Lego, YouTube a tak dále. Jeho silnou stránkou je, že používá TypeScirpt, a tím pádem umožňuje snadnou implementaci OOP, je multiplatformní a má velkou komunitu lidí, jež neustále pracují na jeho zlepšení v každém ohledu. Jednou z nevýhod může být slabší výkonnost, ale ta je v dnešní době velmi výkonných zařízení poměrně zanedbatelná. Můj hlavní důvod pro selekci tohoto frameworku bylo doporučení zaměstnavatele a také jednoduše to, že jej lze snadno zkompilovat jak na iOS, tak na Android a díky tomu jsem mohl vyvíjet na obě platformy najednou.

3.2 Ionic

Ionic je open source SDK pro vývoj hybridních aplikací. Původní vydání v roce 2013 bylo postaveno na AngularuJS a Cordově. Nyní je už však možné vybrat si nejen Angular, ale i další frameworky, a to například React nebo Vue.js. Lze v něm najít UI komponenty, ovládání gesty, API pro animace, ikonky a další. Je velkou pomůckou pro stylizaci aplikace a také samotnou integraci na mobilní zařízení. Umožňuje také testování aplikace pomocí kompilace na localhost či jinou dostupnou IP.

3.3 Apache Cordova

Cordova je framework pro vývoj mobilních aplikací vytvořeno firmou Nitobi. Ta byla později odkoupena společností Adobe Inc. Celý framework slouží k tomu, aby byli vývojáři schopni vytvářet mobilní aplikace s pomocí pouze JS, HTML a CSS. Fungují tak, že Cordova aplikace obalí a vytvoří z nich WebView. Pomocí pluginů také umožňuje využívat v tomto prostředí i mobilní funkce, ke kterým se jinak vývojáři nedostanou. Mezi ně patří například senzory telefonu, kamera, GPS a podobné.

3.4 HTML

HTML neboli Hypertext Markup Language je značkovací jazyk sloužící k tvorbě webových stránek. Stránka se pomocí něj vytváří pomocí tagů, které stránce dávají jakýsi systém a také ji formátují. Často se pojí ještě s kaskádovými styly, díky nimž jsme schopni tyto sekce stránky dále formátovat a upravovat.

3.5 SASS

Sass (Synactically Awestome Style Sheets) je nejnovější rozšíření již poměrně starého CSS. Rozšiřuje ho o proměnné, podmínky, cykly. Já využíval syntaxi SCSS, která je jednostranně kompatibilní se starým CSS. Soubory typu .scss neumí prohlížeč samostatně spustit, a proto je nutné je prohnat právě Sass kompilátorem, který má již Angular v sobě a dělá tuto práci za nás.

3.6 Mapbox

Mapbox je developerská platforma používaná napříč mnoho odvětvími pro tvorbu vlastních aplikací, které řeší problémy s mapami, daty a prostorovou analýzou. Další značnou výhodou je, že každou mapu lze stylově upravit do sebemenších detailů a tím dát své aplikaci jistou odlišnost od ostatních. Jako samotný mapový podklad využívá OpenStreetMap. Jelikož má aplikace je napsaná v TypeScriptu a ten není Mapboxem 100 % podporován, tak jsem musel použít ne příliš známé řešení, a to je ngx-mapbox-gl. Toto je "obal" na původní JavaScriptovou knihovnu vytvořený tak aby šla použít i v Angularu.

3.7 Chart.js

Jak již z názvu vypovídá, jedná se o JavaScriptovou knihovnu určenou pro tvorbu grafů webových aplikací. Uživatelům nabízí až 8 různých typů grafu, animace, a snadnou práci s daty, a to vše jen ve velmi malé knihovně, jež zabírá pouhých 11 kB. Pro Angular opět existuje obalená verze, jež se nazývá ng2-charts, a právě tu jsem použil já.

3.8 Android studio

Je to vývojové prostředí určené k vývoji Android aplikací. Google jej představil v roce 2013 a je neustále aktualizováno. Samotné IDE je velmi robustní a nabízí uživateli velmi mnoho funkcí jako například testování kompatibility, podepisování aplikací, editor vzhledu aplikace s funkcí drag-and-drop atp. Aplikace se v něm tvoří v Javě nebo Kotlinu, vzhled se upravuje pomocí XML. Moje využití tohoto IDE spočívalo ve využití Android emulátoru, díky němuž jsem mohl svou aplikaci testovat na různých zařízeních a různých verzích androidu. Také jsem díky němu vygeneroval klíč k podepsání aplikace.

3.9 Putty

Jednoduchý SSH a Telnet klient pro Windows. Využíval jsem ho k přístupu do příkazové řádky serveru.

3.10 WinSCP

Open source klient umožňující SFTP, FTP vyvinutý českým vývojářem. Tohoto klienta jsem využíval k nahlédnutí do souborového systému serveru a ke stahování a nahrávání dat.

4 Vývoj

4.1 Příprava projektu

Po tom, co jsem už měl ujasněno, jaký je cíl, a které technologie použiji, bylo na čase se dát do práce. Celá další kapitola bude popisovat tvorbu jednotlivých částí celého projektu a jak společně zapadaly do sebe. Začneme tím jednodušším a postupně se propracujeme k složitějším úkolům, u kterých se rozepíšu více aby bylo jasné, jak jsem postupoval.

4.1.1 Výběr vhodného vývojového prostředí

Je zřejmé, že je nutné si i určit v jakém IDE budu projekt vytvářet. Věřím, že TypeScript a JavaScript se dají psát i v poznámkovém bloku, avšak existují i daleko propracovanější textové editory, které jsou také zdarma a velmi usnadní práci například napovídáním syntaxe, zvýrazňováním atp. Já si vybral Atom, jelikož s ním mám předchozí zkušenosti, je zdarma, existuje pro něj TypeScript balíček, jež mi pomohl se orientovat v kódu.

4.1.2 Zjednodušení přístupu k serveru

Další věc, která mi měla usnadnit práci na serveru bylo SSHFS. Bylo pro mě důležité, aby Atom měl přístup na server jako by to byla pouhá složka v mém systému. Díky tomu jsem si mohl vždy projekt otevřít a pracovat v něm. Všechny soubory na serveru se automaticky upravovaly bez nutnosti toho, mít otevřené například WinSCP a upravovat každý po jednom, což by mě velmi zpomalovalo a nejspíše způsobovalo další problémy kvůli oprávněním.

4.1.3 První krok

Jeden z prvních úkolů, jež mi byl přidělen, bylo seznámit se serverem, na kterém budu vše vytvářet. Je na něm nainstalováno Ubuntu s klasickou příkazovou řádkou bez GUI. Na to, že musím dělat vše přes příkazy, jsem nebyl úplně zvyklý, avšak základy Linuxu jsem uměl už z více předmětů jako například POS, PD a tak dále. Provedl jsem tedy instalaci Angularu příkazem:

npm install -g @angular/cli

Jak vidíte, Angular se instaluje přes node package manager (npm), a tak se také instalují všechny knihovny, které budu používat. Npm již bylo součástí mého linuxového balíčku, ale pokud není, dá se naistalovat takto:

sudo apt install nodejs

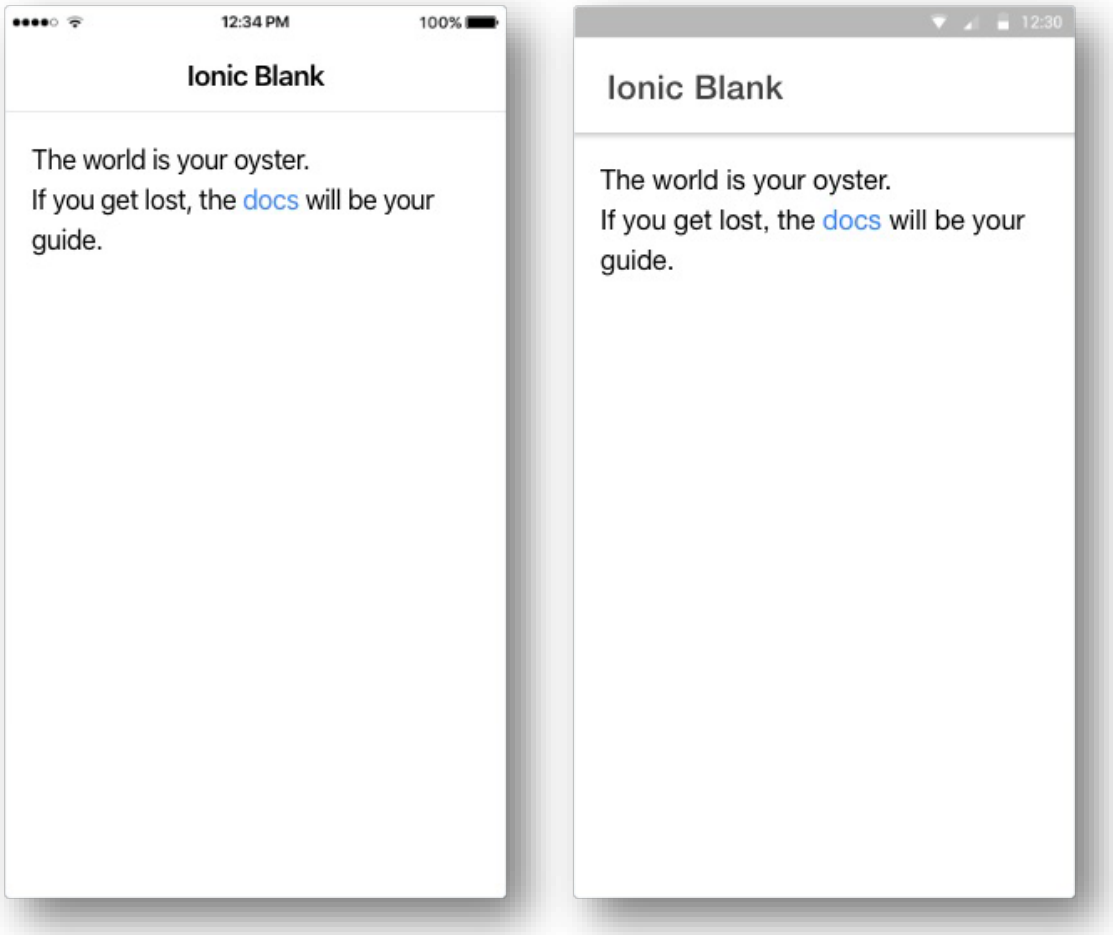
Jako další krok bylo ještě nutno doinstalovat Ionic:

Npm install -g @ionic/cli

Tímto máme na serveru vše, co je potřeba k vytvoření nového projektu, a to provedeme příkazem:

ionic start urbidoAir blank --type=ionic-angular

Tento příkaz vytvořil nový prázdný projekt, jež už má v sobě zakomponovány knihovny Ionicu a Angularu.



Obrázek 3: *Aplikace*

4.2 Implementace Menu

První složitější úkol, jež jsem dostal, bylo vytvoření menu. Představa byla taková, že toto menu bude na boku stránky a bude vysunovatelné a zasunovatelné pomocí tzv. "swipe" akcí. To znamená, že pokud uživatel prstem přejede z levého kraje displeje k druhému kraji, menu se vysune a opačnou akcí zasune zpět, aby nepřekáželo v již obecně malém prostoru, které mobilní displeje nabízí. Ionic vám v tomto může pomoci, pokud vytvoříte nový projekt s přepínačem sidemenu. Toto menu je sice funkční, ale moc bych ho pro vývoj aplikace na míru nedoporučoval, protože vám celý projekt upraví do podoby, jež je tímto menu předem daná a ochudí vás o možnosti si projekt rozvrhnout sám. Užitečné ale určitě je pro to, aby začínající vývojář pochopil, jak takové menu vlastně v Ionic funguje a jak se implementuje. Projekt s tímto předpřipraveným menu lze poté vytvořit takto:

ionic start prikladmenu sidemenu --type=ionic-angular

Já se však vydal cestou menu implementovat sám do našeho prázdného projektu. Jak už tedy ze struktury Angularu vyplývá, musel jsem pro menu založit novou stránku příkazem:

ionic generate page menu

Tento příkaz vytvořil v projektu novou "stránku" aplikace, jež je vlastně složka obsahujících 5 souborů. Každý z nich plní jinou funkci, ať už je to například stylizace nebo samotný backend. To samotné nám ale menu nevytvoří. Bylo nutné ještě dodat kód, který vše zprovoznil. Začal jsem tedy v backendové části, kde jsem vytvořil proměnnou routes. Ta má za úkol sloužit jako zdroj dat pro Angularovský router. Také jsem vytvořil proměnnou pro zrovna vybranou cestu, kde se uživatel nachází.

```
pages = [
  {
    icon: 'home-outline',
    title: 'Úvod',
    url: '/menu/home'
  },
  {
    icon: 'map-outline',
    title: 'Mapa',
    url: '/menu/first'
  }
  ...
];
selectedPath ='';
```

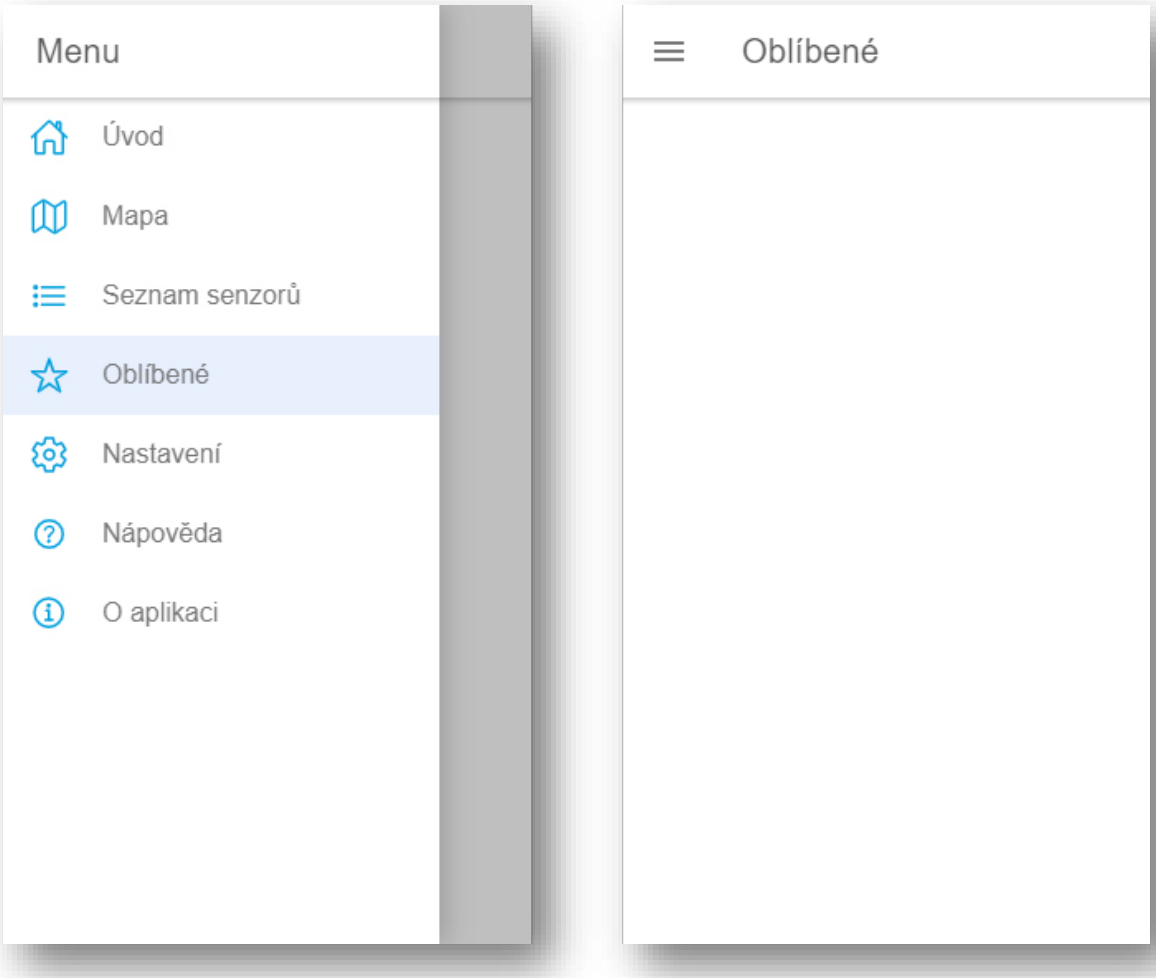
Jak lze z kódu vyčíst, každá stránka je tedy reprezentována ikonkou, názvem a url adresou, kde se stránka nachází. Tyto informace pak je nutné zpracovat dále v HTML. Ve stránce s kódem jsem ještě také musel doimplementovat samotný router, který bude můj objekt s cestami k samotným stránkám využívat. To jsem provedl takto:

```
constructor(private router: Router) {
this.router.events.pipe(filter(e => e instanceof
NavigationEnd))
    .subscribe((e: NavigationEnd) => {
      this.selectedPath = e.url;
    });
}
```

Jelikož není Router součástí jádra Angularu, tak bylo nutné ho zahrnout i do konstruktoru. Celá tato navigace funguje na základě eventu routeru, jež se vyvolá v HTML při kliknutí na danou stránku, a router pak přesměruje uživatele pomocí selectedPath proměnné, do které se data z eventu vyfiltrují.

```
    <ion-content>
      <ion-list>
        <ion-menu-toggle *ngFor="let p of pages">
          <ion-item  [routerLink]="p.url" routerDirection="root"
[class.active-item]="selectedPath === p.url" lines="none">
          <ion-icon class="icon-margined" name={{p.icon}}>
          </ion-icon>
          {{p.title}}
          </ion-item>
        </ion-menu-toggle>
      </ion-list>
    </ion-content>
  </ion-menu>
```

Jak si lze všimnout, použil jsem *ngfor, což je vlastně cyklus v HTML. Ten vytvoří jednu položku v menu pro každou stránku v objektu pages. Na každou je pak aplikován router link a ikona ze souboru ikon Ionic5. Tím je menu hotovo.



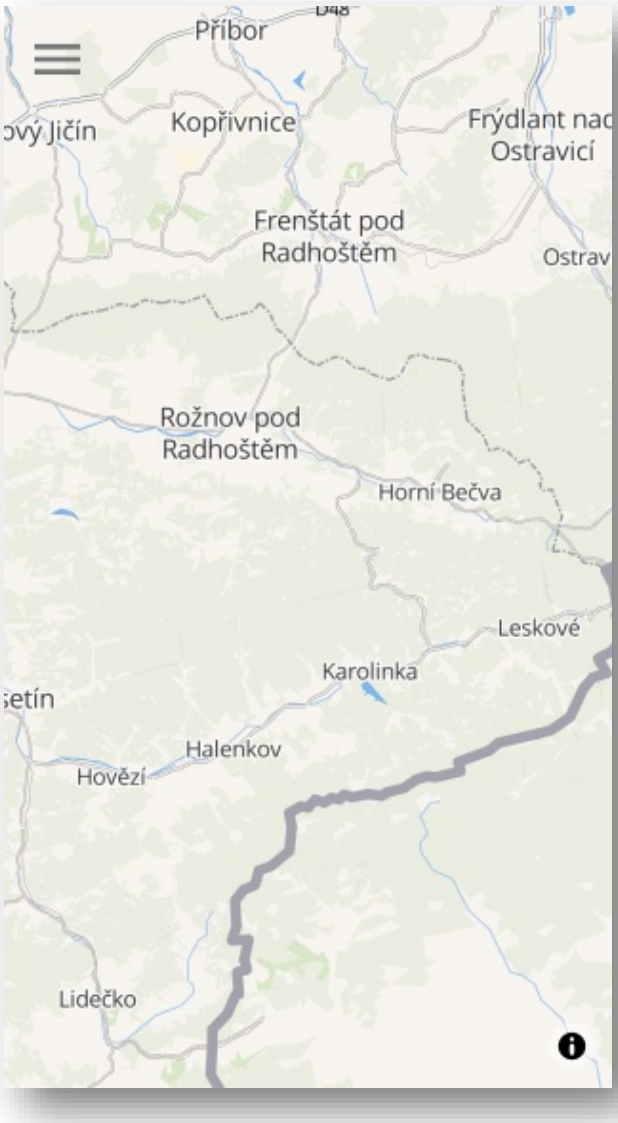
Obrázek 4: *Menu aplikace*

4.3 Implementace Mapy

Jako další z úkolů mi bylo zadáno do aplikace implementovat interaktivní mapu. Představa byla taková, že na obrazovce bude uživateli zobrazena mapa, na které se budou nacházet jednotlivé hodnoty senzorů po celé České republice. Hodnoty měly být zobrazeny v jakémsi kruhu, jehož barva bude záviset na současné snímané hodnotě AQI. Po kliknutí uživatele, na jakýkoliv z těchto senzorů, se měla otevřít nová stránka, jež bude věnována jenom tomuto konkrétnímu senzoru. Další požadavek byl, aby se senzory tzv. "clustrovaly", tj. že při dostatečném oddálení mapy nechceme vykreslovat jeden kruh přes druhý, ale nýbrž všechny dotýkající se zprůměrovat a vykreslit místo nich kruh nový, jež bude zobrazovat právě zprůměrované hodnoty všech senzorů do sebe absorbovaných. Jako mapu jsem měl využít předešle zmíněnou OpenStreetMap upravenou společností Mapbox.

Z důvodu, že Mapbox neposkytuje podporu pro TypeScript a mapa je hlavní bod celé aplikace, bylo nutné promyslet, jak ji v mém prostředí vlastně zprovoznit. Následovalo tedy extenzivní vyhledávání možných řešení. Nakonec jsem se rozhodl jako řešení využít projekt ngx-mapbox-gl, což je projekt malého týmu lidí, vytvořit obal pro mapbox-gl-js, který by fungoval na Angularu. Tento projekt je volně dostupný na GitHubu. Musel jsem tedy na server nainstalovat jeho knihovny, tj. ngx-mapbox-gl a ještě samotné knihovny mapbox-gl.

Po tomto procesu jsem byl připraven na implementaci, a tak jsem postupoval podle návodu na GitHubu, abych vytvořil onu prázdnou mapu, což proběhlo bez problému.



Obrázek 5: *Prázdná mapa*

4.3.1 Doplnění mapy o senzory

Mapa jako taková byla tedy implementována. Teď ale nastal jeden z nejtěžších úkolů celé práce, a to bylo na této mapě zobrazit data ze senzorů. Jak už jsem řekl, celý "wrapper" je jen malý projekt, jež ještě nemá implementovány všechny původní funkce, které Mapbox nabízí pro JS, a tak implementace zadání v tomto případě byla dosti obtížná. V samotném kódu jsem totiž byl schopen jen nastavit základní věci, jako je například centrum mapy, její zoom atp. Nikoliv zobrazení samotných senzorů a clusterů. Toto mě dovedlo k rozhodnutí udělat jako první věci, které se dají napsat čistě v TypeScriptu a až poté se vrhnout na další, které pro mne byly zatím nejasné.

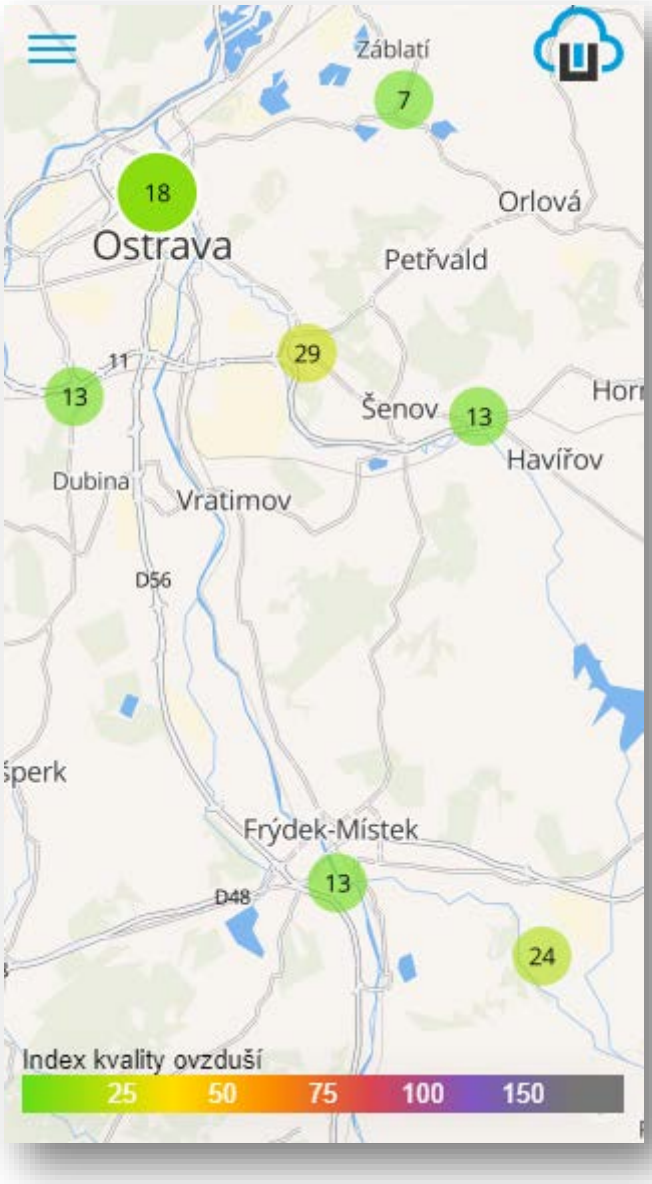
To zahrnovalo:

- Zjištění, zda je uživatel připojen k internetu
  - Funkce, která zjišťuje, jestli má celé načítání vůbec smysl, jelikož mapa bez připojení k internetu není schopná nic zobrazit, a tedy vše dále nemá smysl.
- Stažení všech dat senzorů ze serveru
  - Pokud připojení k internetu existuje, pak pomocí metody GET stáhnout všechna aktuální data senzorů jako JSON objekt do aplikace.
- Ověření, zda má uživatel povolenu geolokaci
  - Pomocí Local Storage zjistíme, zda uživatel povolil aplikaci využít jeho polohu, aby se mohla mapa zacentrovat na místo, kde se uživatel nachází.
- Získání uživatelovy polohy
  - Pokud uživatel povolil geolokaci, pak se jeho souřadnice nahrají místo výchozích jako centrum mapy.
- Přednastavení
  - Nakonec se ze zjištěných dat celá mapa přednastaví a je zobrazena uživateli.

Po dalším zkoumání mě napadlo vykreslování hodnot senzorů vyřešit v samotném HTML souboru. To vycházelo z toho, že funkce pro TypeScript byly značně nedostačující a jiné řešení dodnes neexistuje. V HTML lze totiž po instalaci knihovny využít prvek <mgl-map> a také <mgl-geojson-source>. Pomocí prvního prvku a jeho sub-prvku <mgl-layer> lze vykreslit do mapy bod, jež jde pomocí značek nastavit tak, aby vyhovoval zadání a zároveň na něj jde aplikovat metoda "onClick", díky níž bude možné se přepnout na soukromou stránku senzoru. Abych získal data, jež budou tyto vrstvy používat užil jsem druhého prvku, který rozkouskuje předem získaný JSON objekt s daty o senzorech a poskytne ho mapě. Kód jsem doplnil ještě o metody onClick a onClickCluster.

- onClick – je metoda, jež se volá, když uživatel klikne na senzor. Po kliknutí uživateli zobrazí novou stránku věnovanou jen vybranému senzoru.
- onClickCluster – je metoda volající se při kliku na cluster. Po kliknutí přiblíží místo, ve kterém se cluster nachází, aby mohl uživatel vidět jednotlivé senzory.

Stylově jsem do mapy také dodal samotné logo firmy a legendu pro uživatele.



Obrázek 6: *Hotová mapa*

Jak lze z obrázku vidět, cluster se jeví jako plné kolečko s bílým tenkým okrajem, kdežto senzor je kolečko o moc menší a také jemně průhledné. Čísla, jež lze v kruzích pozorovat se nazývají AQI. Na spodní části snímku obrazovky lze vidět legendu kvality ovzduší, kdy číslo menší znamená lepší ovzduší a méně částic prachu.

4.4 Implementace vlastního okna senzoru

Dalším úkolem mi bylo přiděleno udělat mapu responzivní. Současná mapa má sice funkce oddalování, přibližování posouvání atd., avšak uživatel se nedozví víc než hodnotu v okolí vyznačeného senzoru. Neví jeho název, jeho reálné hodnoty PM10 a PM2.5, jeho historii. Bylo tedy ideální, aby se uživatel dozvěděl více pokud na některý ze senzorů klikne. Tento úkol jsem tedy realizoval postupně na dvě pod stránky, kdy na první se nachází obecné informace a na druhé graf s historií hodnot.

Prvním krokem k vytvoření těchto stran aplikace bylo samotné zajištění dat. To bylo poměrně jednoduché, jelikož stačilo pouze vytvořit GET dotaz na server, jež mi odpověděl JSON objektem, což byl samotný senzor i se všemi jeho hodnotami. To zahrnuje jeho lokaci, AQI, PM10, PM2.5 a datum poslední aktualizace.

Po získání dat bylo nutné tyto data "rozparsovat", tj. přeměnit tento string dat do nějakého objektu, z kterého pak programaticky mohu čerpat data s zobrazovat je uživateli. Jako bonus jsem referenční barvu kvality vzduchu zavedl i na tuto stránku a přidal i slovní hodnocení současného stavu.

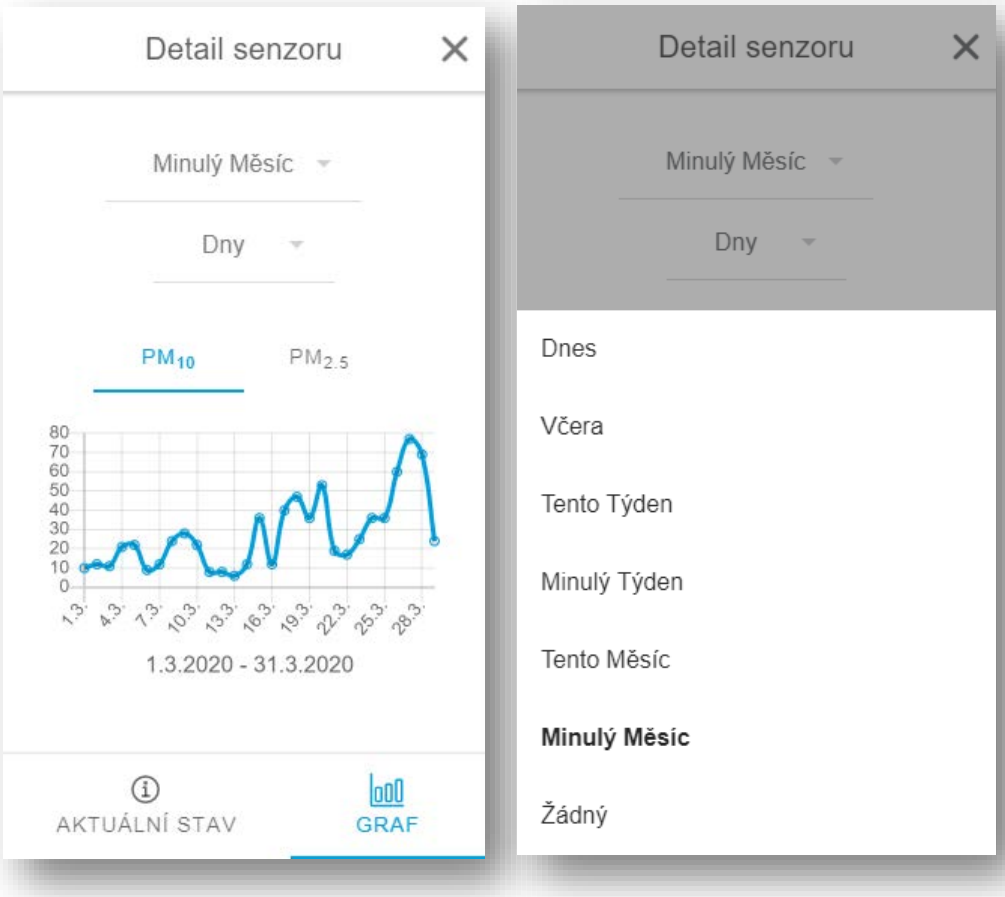
Obrázek 7: *Detail senzoru*

V druhé části jsem se zabýval samotným grafem. Uživatel se na tuto stránku přepne stisknutím prvku tabulátoru ve spodní části obrazovky. Graf tvoří právě výše zmíněný Chart.js upravený pro použití v Angularu. Chart.js jako takový umí spoustu typů grafů, já využil graf typu lineChart. Objekt grafu je opět nutné vyplnit daty, které má graf zobrazit, a tudíž bylo nutné opět požádat databázi na serveru o data nutné k vykreslení. Nemohu však zobrazit vše v jednom, jelikož by takové zobrazení bylo matoucí, a tak jsem vytvořil sadu ovládacích prvků, kde si uživatel může navolit jaké období chce zobrazit a jak podrobně.

V prvním selektoru lze nastavit tyto možnosti: dnes, včera, tento týden, minulý týden, tento měsíc, minulý měsíc.

V druhém selektoru lze nastavit, zda zobrazit data hodinově či po dnech. Možnost dnů se však zobrazí pouze pokud v prvním selektoru vybereme období týden a delší.

Třetí selektor umožňuje výběr mezi částicemi PM10 a PM2.5, pokud senzor měří obě hodnoty.



Obrázek 8: *Detail senzoru - graf*

4.5 Implementace listu senzorů

Aby uživatel mohl senzory v aplikaci procházet i mimo mapu, bylo vhodné naimplementovat jako jednu z položek menu samotný list všech senzorů, kde by uživatel mohl přehledně sledovat naměřené hodnoty, aniž by musel brouzdat mapou a hledat jejich přesné pozice. Jako základní filtr bude použita geolokace, která seřadí položky podle vzdálenosti vzdušnou čarou mezi uživatelem a měrným zařízením.

Jako první bylo nutno vyřešit vytvoření samotného listu a naplnit ho daty. Vytvořil jsem si tedy objekt objektů senzorů. Do jiného objektu si pak vždy stáhnu data ze serveru a postupně je roztřídím mezi jednotlivé senzory. Když je vše toto připraveno lze poté do HTML části použít Angular tag *ngfor, který mi objekt rozbalí a ze samotných jeho částí vytvoří položky listu. Na tomto principu už fungovalo menu aplikace, takže jsem byl s tímto principem už dobře seznámen.

Nyní jsem potřeboval zprovoznit geolokaci. Jelikož je při tomto kroku nutné zjistit polohu zařízení, musel jsem využít Cordovy. Díky ní můžeme využívat funkcí mobilních zařízení jinak nepřístupných. To zahrnuje věci jako například notifikace, geolokaci, gyroskop a jiné. Instaluje se pomocí CLI tímto příkazem:

cordova plugin add cordova-plugin-geolocation

Po instalaci a implementaci do projektu je možné používat funkci getCurrentPosition(), jež vrátí zeměpisnou šířku a délku uživatele. Ty jsem poté společně se zeměpisnou šířkou a délkou samotného senzoru zaslal do funkce která mi vrátí vzdálenost dvou bodů v km.

```
public getDistanceInKm(lat1,lon1,lat2,lon2) {
 var R = 6371;
 var dLat = this.deg2rad(lat2-lat1);
 var dLon = this.deg2rad(lon2-lon1);
 var a =
     Math.sin(dLat/2) * Math.sin(dLat/2) +
     Math.cos(this.deg2rad(lat1)) *
     Math.cos(this.deg2rad(lat2)) *
     Math.sin(dLon/2) * Math.sin(dLon/2);
     var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
     var d = R * c;
     return Math.round(d*100)/100;
 }
```

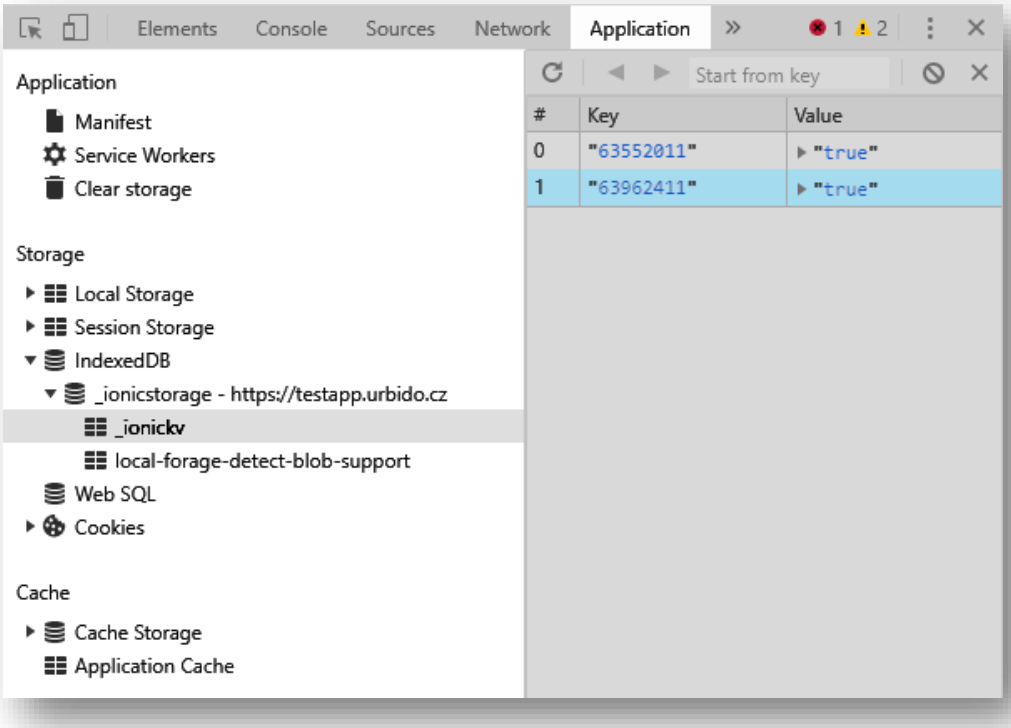
Tato funkce je vlastní implementace Haversinovy formule, která je běžně používána v navigačních systémech. Výsledek však není 100 % přesný, jelikož rovnice počítá s tím, že Země je perfektní koule.

$$\boldsymbol{d = 2r\arcsin(\sqrt{sin^2\left(\frac{\emptyset_2-\emptyset_1}{2}\right) + \cos(\emptyset_1)\cos(\emptyset_2)\,sin^2\left(\frac{\psi_2-\psi_1}{2}\right)})} \tag{4.1}$$

Kde **d** je vzdálenost dvou bodů [km], **r** je střední poloměr Země [6371 km], $\boldsymbol{\Phi_1, \Phi_2}$ je zeměpisná šířka určených bodů, [◦], $\boldsymbol{\Psi_1,\Psi_2}$ je zeměpisná délka určených bodů [◦]

V neposlední řadě bylo nutné vložit ke každému prvku hvězdičku reprezentující oblíbený senzor. Napadlo nás, že by pro uživatele bylo přívětivé, mít možnost některé ze senzorů označit jako oblíbené a věnovat jim poté svou samostatnou stránku, na které by uživatel viděl pouze pár jeho vybraných favoritů, nikoliv celý seznam desítek možných zařízení. Díky tomuto stačí navštívit samotný dlouhý list pouze jednou, zalistovat v něm a vybrat si jen ty, které ho opravdu zajímají. V dalším navštívení aplikace budou tyto favorizované měřiče, již k dispozici na separátní stránce.

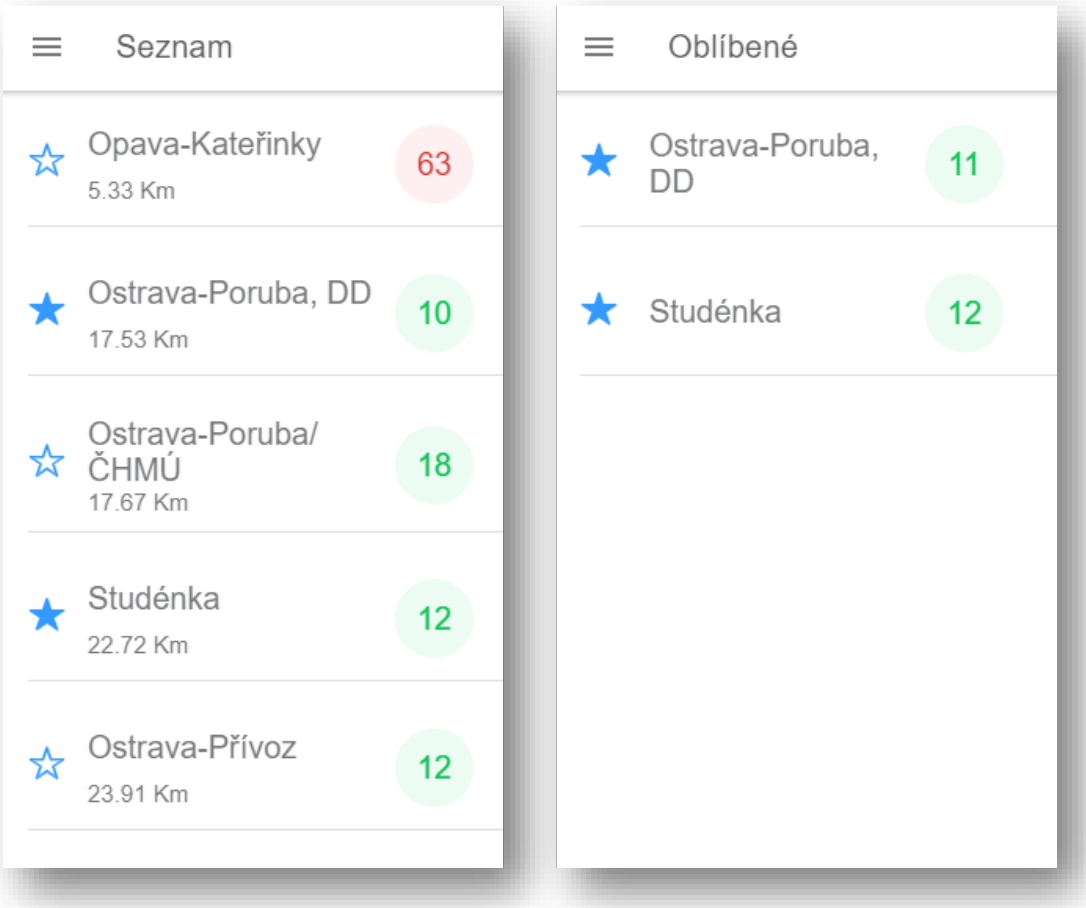
Toto jsem do aplikace přidal využitím localStorage. V tomto ohledu mi pomohl Ionic, protože umí využít tuto funkci nejen ve webových aplikacích ale i v těch mobilních. Dodal jsem tedy objektu senzoru boolean prvek hvězdičky, která je v základu nastavená na nepravda. V cyklu pak projedu všechny klíče uložené v paměti a uložím si jejich hodnoty. Senzorům, jejichž ID je identické s nějakým z uložených klíčů pak nastavím hvězdičku na hodnotu pravda. Princip



Obrázek 9: *Ionic Storage*

ukládání a mazání je o mnoho jednodušší, kdy jen do Ionic Storage přidám klíč s ID daného senzoru, či jej odtud odstraním. Díky principu ukládání do paměti telefonu, mohu k těmto datům přistupovat odkudkoli v aplikaci a velmi si usnadním práci na další stránce věnované pouze oblíbencům bez toho, aniž bych duplikoval některá data.

V neposlední řadě jsem upravil vzhled celého seznamu pomocí SASS a HTML. Vlevo vidíme znak hvězdy značící, zda je senzor oblíbený či nikoliv, dále pak samotný název a pod ním vzdálenost senzoru od polohy mobilního zařízení. V pravé části obrazovky lze pak naleznout hodnotu AQI s odpovídající barvou vyznačující současnou situaci.



Obrázek 10: *Seznamy senzorů*

Ovládání všech prvků seznamu jsem se snažil udělat intuitivní. To znamená, že jsem se snažil vyhnout jakýmkoli posuvníkům a podobným prstem těžko uchopitelným, či stisknutelným prvkům.

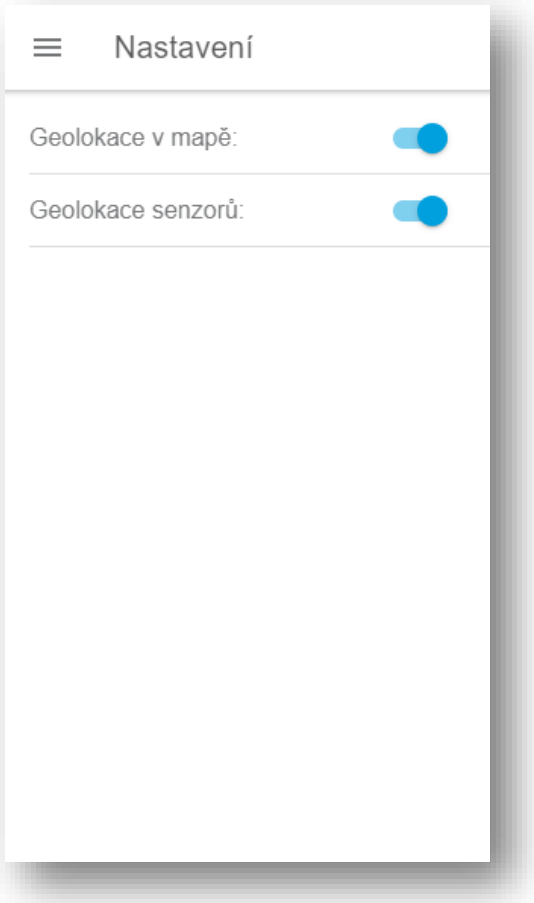
V seznamu se posouváme nahoru a dolů pomocí tahu prstem. Pokud uživatel klikne kdekoliv do oblastí oddělených šedými čarami vyjma hvězdiček, bude přesunut na stránku daného odpovídajícího měřiče vytvořenou v předchozích krocích práce. Z ní se dostane akcí "swipe", tj. přetáhnutí prstu z levého kraje obrazovky vpravo, zpět na seznam. Je možné použít i křížek v pravém horním rohu obrazovky viz. Obr. 7.

Pokud systém zaznamená klik na hvězdičku, senzor, jemuž patří bude uložen do paměti a při návštěvě položky "Oblíbené" zobrazen zde. Hvězdičku lze odkliknout v jakémkoli ze seznamů, tím zmizí její výplň jako indikace této akce a oblíbený měřič bude odstraněn z paměti. Volbu je možné si rozmyslet zakliknutím znova, jelikož akce se provádí až po rozhodnutí uživatele navštívit jinou část aplikace.

4.6 Nastavení

Protože jsem se snažil, aby si uživatel nemyslel, že je sledován, naimplementoval jsem do aplikace i stránku nastavení. Zde se dá vybrat, zda se má využívat geolokace při funkci aplikace či nikoliv.

Přidat tuto funkci do aplikace bylo poměrně jednoduché, protože povolení se uchovává interně na zařízení pomocí booleovských hodnot. Jsou zde separátně uloženy jak pro mapu, tak pro list senzorů a obě lze ovládat zvlášť.



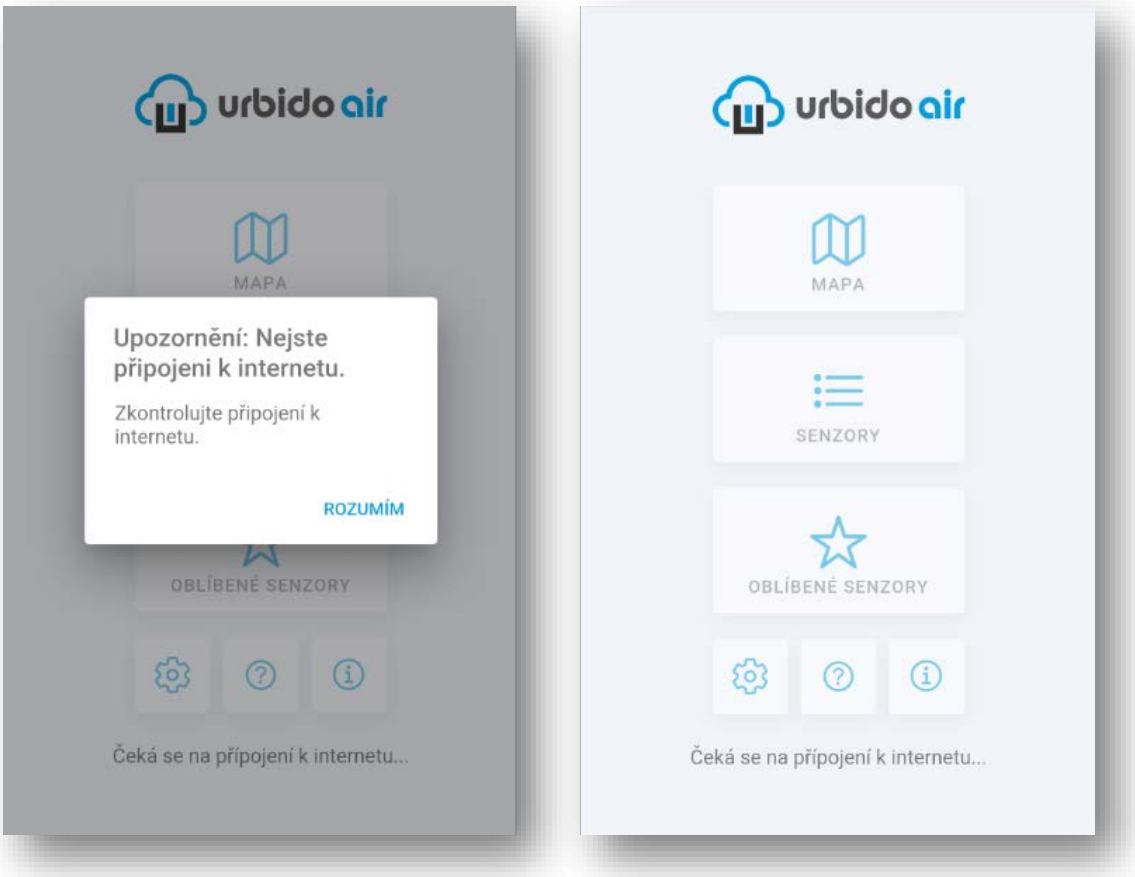
Obrázek 11: *Nastavení aplikace*

Stránka se může zdát na první pohled poměrně prázdná, ale je nutno brát v úvahu, že aplikace se bude časem určitě dále vyvíjet a prostor zde se zaplní.

4.7 Ochrana proti výpadkům internetu

V neposlední řadě mě při vývoji napadlo, že každá aplikace vyžadující připojení k internetu by měla mít jakýsi systém, který omezí uživatelovy funkce v případě, pokud připojení selže. Toto se může zdát z počátku jako něco ne příliš nutného, avšak bez internetu nemá aplikace možnost stáhnout si data, které jsou potřebné k vykreslení mapy, všech senzorů, grafů atp. Pokud by uživatel nevědomě spustil aplikaci bez připojení, byla by aplikace prázdná a funkční sotva z poloviny. Na základě toho by si nejspíše uživatel myslel, že aplikace je nefunkční a odinstaloval by ji ze svého zařízení. Následovalo by negativní hodnocení na Google Play a vydání by tak nemuselo být vůbec úspěšné.

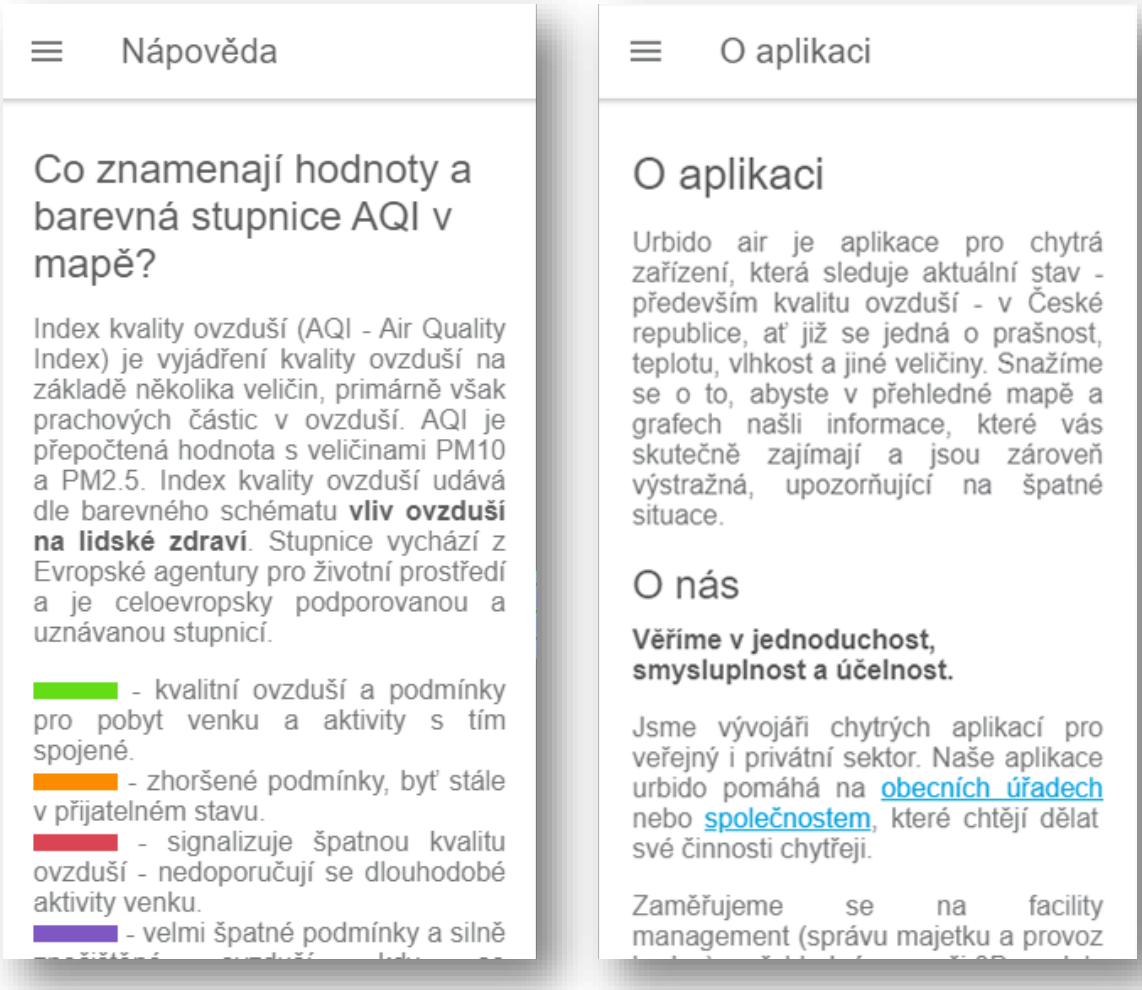
Řešení spočívalo v několika krocích. Prvním z nich bylo využití návrhového vzoru "observer". Existuje tedy jakýsi pozorovatel, zapnutý při startu aplikace, jehož jediný úkol je sledovat připojení k internetu. Běží nad jakoukoli stránkou aplikace a při změně stavu připojení jí okamžitě informuje, a nutí její router k přepnutí na úvodní stránku, kterou bylo také nutné udělat. Jako extra krok jsem zvolil přidání oznámení, které jasně uživateli oznamuje změnu stavu internetového připojení. Z obrázku také jde vidět, že pokud uživatel odklikne upozornění a stále nemá připojení k internetu, tlačítka jsou zašedlá a nefungující.



Obrázek 12: *Ochrana proti výpadku internetu*

4.8 Další stránky

Samotná aplikace se skládá současně ještě z dalších dvou stránek, které sice nebyly nijak těžké vytvořit, ale protože pointa této práce je nastínit, jak asi vypadá vývoj takové mobilní aplikace z nuly až po vydání, je podle mě vhodné, aby zde byla zobrazena celá. Tyto dvě stránky slouží jako zdroj informací pro uživatele. Může se zde dočíst něco o firmě urbido s.r.o. a třeba si i objednat senzor anebo se trošku víc zorientovat v problematice znečištěného ovzduší a pochopit tak, co vlastně všechny čísla a hodnoty v této aplikaci znamenají.

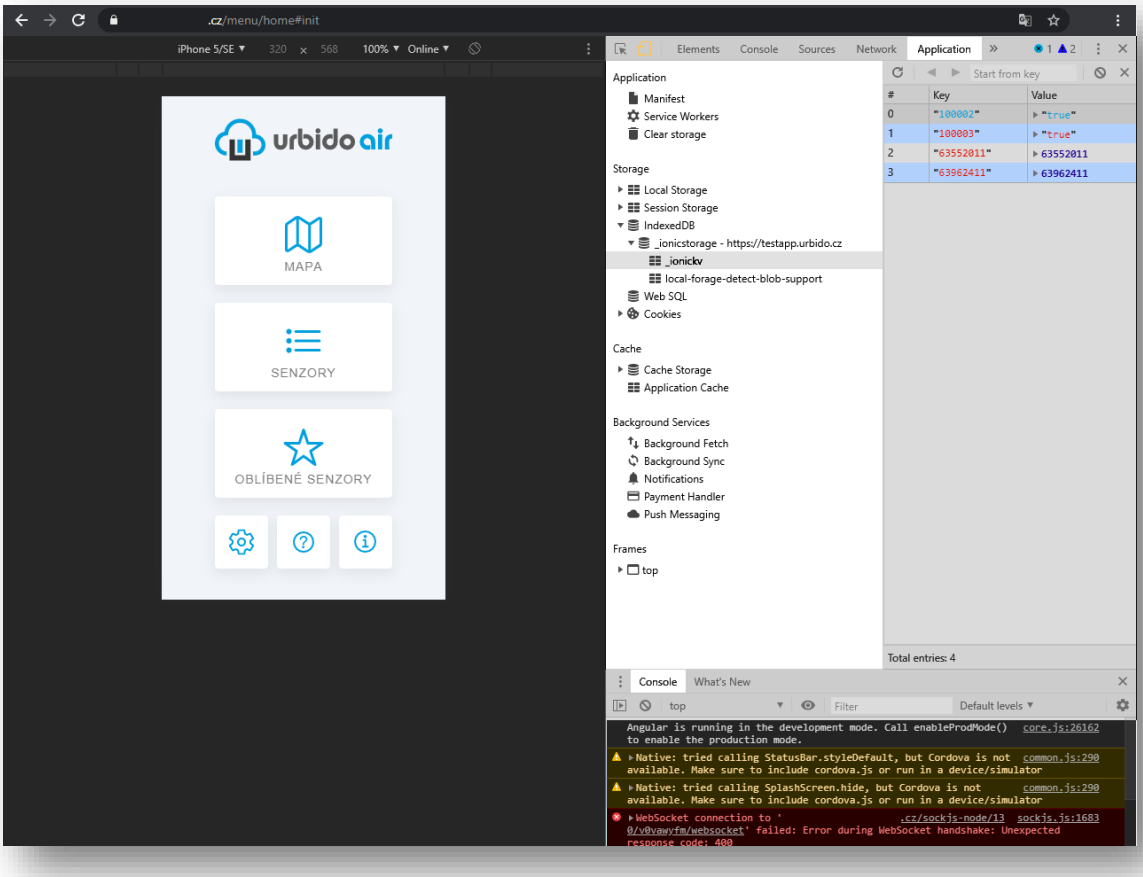


Obrázek 13: *Informace*

5 Testování

V neposlední řadě bych se ještě rád zaměřil na téma testování. Jak jsem se už na začátku práce zmínil, tak samotný Ionic dovoluje kompilaci a testování na localhost adresu nebo jinou přístupnou IP specifikovanou v příkazu. Tuto funkci jsem pro testování využíval velmi často, hlavně z počátků, kdy aplikace ještě neobsahovala žádné prvky Cordovy, které vyžadovaly reálné zařízení. Mezi výhody tohoto testování patří určitě jeho rychlost, možnost testovat na různé velikosti zařízení a nahlížení do paměti zařízení. Výhodou je také přítomná konzole, díky které můžeme odhalovat chyby.

Další ze stylů testování je samotné Android studio a jeho emulovaný mobilní telefon. Zde už lze zkoušet funkce naimplementované pomocí Apache Cordova, avšak již nemáme šanci se dostat k datům v paměti či samotné konzoli. To vyplývá z toho, že jsme aplikaci již převedli na .APK a nemáme přístup ke zdrojovému kódu. Sečteno podtrženo, ke správnému otestování je potřeba využívat obě možnosti a k nim přidat ještě co nejvíce reálných telefonů se systémem Android, nejlépe s různými verzemi.



Obrázek 14: *Testování na webu*

6 Zveřejnění na Google Play

Po úspěšném otestování všech funkcí byla aplikace připravena k vydání na Google Play. Pro vydání na iOS jsme se rozhodli ještě počkat, protože nebyl dostatek času k testování a také jsme si úplně nebyli jistí procesem schvalování aplikací na App Store.

Vedení firmy tedy zařídilo všechny administrativní záležitosti a na mně zbylo připravení samotné aplikace. To se skládalo z několika kroků.

Začneme tím, že samotnou aplikaci označíme jako "release" verzi a vytvoříme ji pomocí Cordovy.

cordova build --release android

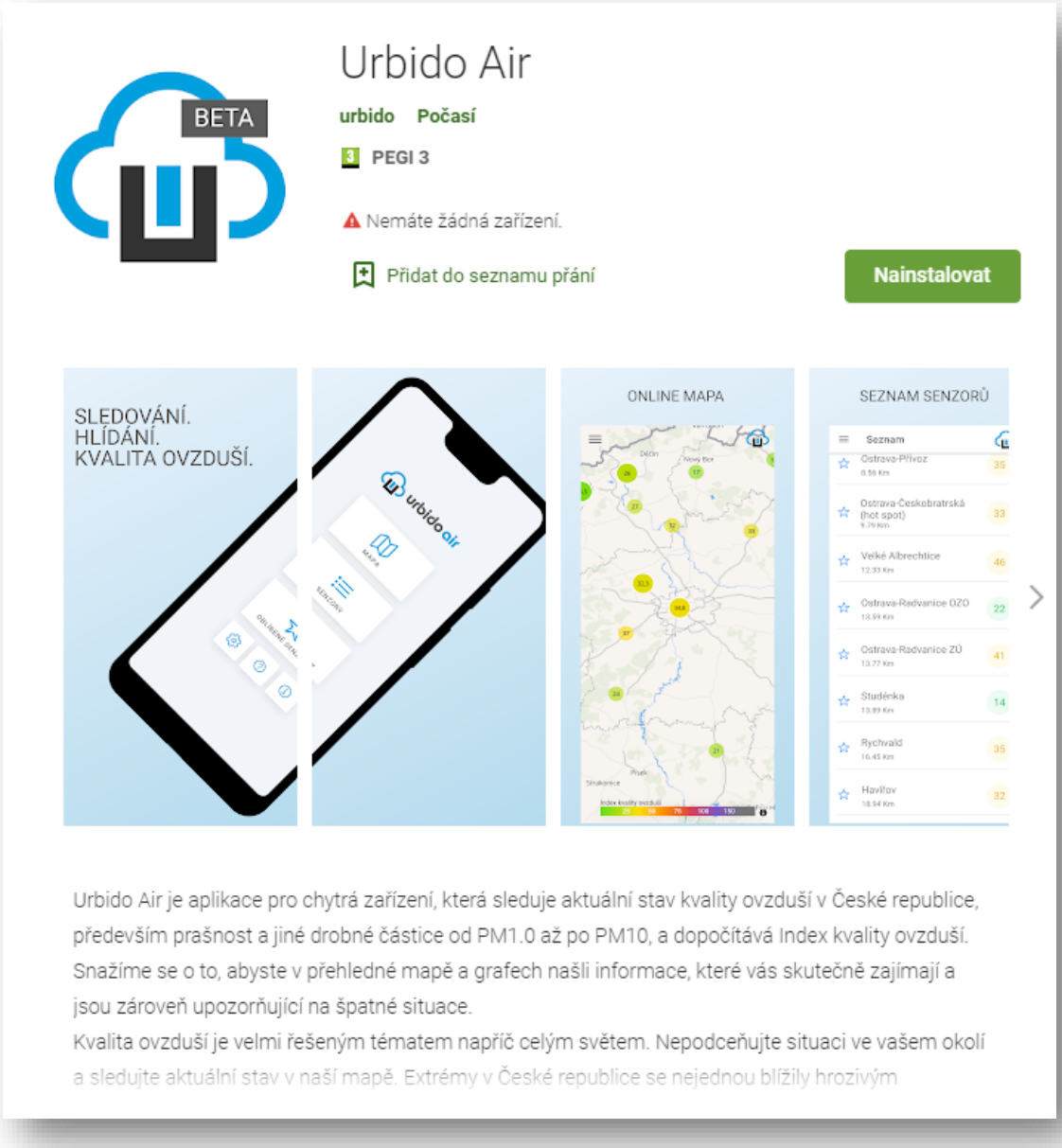
Poté musíme využít funkcí Android studia a vytvořit si "self-signed" certifikát pro naši aplikaci.

keytool -genkey -v -keystore aplikace.keystore -alias certifikat_aplikace -keyalg RSA -keysize 2048 -validity 10000

Nejdůležitější je nastavit správnou hodnotu validity, která nám říká kolik dní bude certifikát platný. Po jeho ukončení se musí generovat nový a aplikace už by se nemohla aktualizovat. Proto zvolíme délku například 10 000 dní, což zajistí, že aplikace bude validní dokonce svého života. Tento klíč je nutné si někde uschovat, protože bude u každé aktualizace požadován. Zbude nám tedy ještě jeden poslední krok, a to je využít aplikaci Zipalign, která poskytne aplikaci požadovanou optimalizaci – významně sníží množství použité RAM při běhu.

zipalign -v 4 aplikace-neoptimalizovana.apk aplikace-optimalizovana.apk

Po těchto krocích byla aplikace připravena k nahrání na Google Play. Už stačilo jen potvrzení samotného Googlu a uživatelé mohli začít stahovat.



Obrázek 15: *Aplikace na Google Play*

7 Uplatněné a chybějící znalosti

7.1 Uplatněné znalosti

Mezi nejdůležitější znalosti nabyté ve škole bych určitě vyzdvihl předměty Tvorba aplikací pro mobilní zařízení 1 a 2, ve kterých sem se nejen seznámil s JavaScriptem, který je TypeScriptu zlehka podobný ale i se samotným Android Studiem a jeho mnoha funkcích, bez kterých bych nebyl závěr práce dokončit. Zde jsem také získal přehled o tom, jak se pracuje programově se servery, JSON soubory, XML soubory a podobně.

Pro samostatné kódování se mi hodily znalosti z předmětů Algoritmy I, II a Programování I, II, které mi daly vědomosti o OOP, sort algoritmech, klíč-hodnota systémech a obecně o tom, jak nějaký problém rozkouskovat na mnoho menších, aby byl zvládnutelný.

Pro návrh struktury aplikace jsem využil vědomosti získané v Úvodu do softwarového inženýrství, kde jsem se dozvěděl o návrhových vzorech a jak je využívat.

Dále jsem uplatnil znalosti práce s linuxovým systémem nabrané například v předmětech Přenos dat, Praktikum komunikačních sítí I a jiných.

V neposlední řadě bych ještě uvedl předmět Počítačové sítě, kvůli předaným znalostem o protokolech používaných v rámci internetu.

7.2 Chybějící znalosti

Jako hlavní nedostatek bych považoval chybějící informace o TypeScriptových jazycích. Samotné učení Angularu mi totiž zabralo dost času, jak doma, tak i během pracovních hodin ve firmě, protože to byla pro mě věc zcela nová a trvalo mi se v něm zorientovat. V dnešní době, kdy je tento jazyk velice populární by to určitě budoucí studenti ocenili.

Mezi vedlejší nedostatky bych zařadil to, že žádný z programovacích předmětů neměl v osnovách jakoukoliv práci s mapou. Myslím si, že třeba aplikace učiva o grafech spojené s učivem o tom, jak ho poté využít v mapě jako například navigaci by bylo záživnější a opět by to dalo studentům o něco navíc.

V neposlední řadě bych uvítal více znalostí o tom, jak správně tvořit design moderních aplikací. Předmět zabývající se například pouze kaskádovými styly bych uvítal.

Závěr

Celou praxi bych za mne hodnotil jako velmi přínosnou. Díky tomu, že jsem na celém vývoji pracoval jako jednočlenný tým, jsem získal možnost dělat jak frontend tak i backend aplikace, zjistit co mě baví, v čem chci pokračovat a zlepšit se v obou věcech.

Získal jsem také cenné znalosti o tom, jak probíhá vývoj od samotného začátku až do konce, díky nimž budu v budoucí práci schopen se zařadit do jakéhokoliv vývojového stavu projektu s vědomostmi v této části potřebnými. Jako největší plus celé praxe považuji to, že jsem se naučil pracovat s mnoha novými technologiemi, které jsou v současné době moderní a rozšířil se mi tak i životopis a možná nabídka práce.

Zdůraznil bych také užitečnost celého projektu z důvodu důležitosti tématu, kterým se zaobírá. Kvalita ovzduší je v současné době velmi populární téma, přičemž jedním z důvodů je fakt, že zvýšená koncentrace prachu ve vzduchu způsobuje onemocnění plic. Tento projekt může mnoha lidem pomoci začít se orientovat v této problematice a chránit jejich zdraví.

Samostatná konečná aplikace potěšila nejen mě ale i zaměstnavatele, a tak je možné, že na jejím dalším vývoji se budu podílet právě já. Mé přání je, aby byla aplikace populární a mé rozhodnutí vykonat praxi mě posouvalo ještě dál, než mě posunulo dosud.

Pokud bych měl zhodnotit metody použité k dosažení cíle, tak volbu Angularu nelituji, kvůli jeho ladnosti a flexibilnosti. Nevýhodou bylo poměrně dlouhé doučování zmíněného programovacího jazyka. Také pro aplikace vyžadující náročné výpočty, především ty grafické je to nevhodná volba z důvodu omezeného výkonu plynoucího z možnosti vytvářet na více platforem najednou. Volbu MapBoxu jako poskytovatele mapy bych zhodnotil neutrálně z důvodu špatné kompatibility s mou volbou jazyka, kde jediná velká výhoda oproti konkurenčním mapám od společností Google či Apple byla pouze cena. U ostatních knihoven a metod jsem žádné významné nevýhody nezaznamenal.

Použitá literatura

[1] FALTEJSEK, Michal. Urbido logo. In: Wikipedie: Otevřená encyklopedie [online]. 2018 [cit. 2020-05-07]. Dostupné z: https://cs.wikipedia.org/wiki/Soubor:Urbido_logo.png

[2] Autor neznámý. Particulate size comparison. In: California Air Resource Board [online]. [cit. 2020-05-04]. Dostupné z: https://ww2.arb.ca.gov/sites/default/files/inline-images/human_hair_0.png

[3] Your First Ionic App: Angular. Ionic Framework [online]. [cit. 2020-05-04]. Dostupné z: https://ionicframework.com/docs/angular/your-first-app

[4] NNANNA, John. CONVERT YOUR ANGULAR (v7 updated) PROJECT TO MOBILE APP USING CORDOVA. In: Medium [online]. 10 October 2017 [cit. 2020-05-04]. Dostupné z: https://medium.com/@nacojohn/convert-your-angular-project-to-mobile-app-using-cordova-f0384a7711a6

[5] DE JABRUN, Guillaume. Ngx-mapbox-gl. GitHub [online]. 2020-03-02 [cit. 2020-05-04]. Dostupné z: https://github.com/Wykks/ngx-mapbox-gl

[6] AGARWAL, Prakhar. Haversine formula to find distance between two points on a sphere. GeeksforGeeks | A computer science portal for geeks [online]. [cit. 2020-05-04]. Dostupné z: https://www.geeksforgeeks.org/haversine-formula-to-find-distance-between-two-points-on-a-sphere/

[7] Kvalita ovzduší. Urbido Air [online]. [cit. 2020-05-04]. Dostupné z: https://air.urbido.cz/kvalita-ovzdusi

[8] Přispěvatelé Wikipedie. Angular (web framework). In: Wikipedie: Otevřená encyklopedie [online]. 1 May 2020 11:04 UTC [cit. 2016]. Dostupné z: https://en.wikipedia.org/wiki/Angular_(web_framework)

[9] Přispěvatelé Wikipedie. Ionic (mobile app framework). Wikipedie: Otevřená encyklopedie [online]. 9 April 2020 16:36 UTC [cit. 2020-05-04]. Dostupné z: https://en.wikipedia.org/wiki/Ionic_(mobile_app_framework)